Ramblings on Agile Methodologies and Ontology-Driven Software Development

Position Paper for the Workshop on
Semantic Web Enabled Software Engineering, 2005

Holger Knublauch
holger@smi.stanford.edu

Stanford Medical Informatics, Stanford University, Stanford, CA
Medical Informatics Group, The University of Manchester, UK

The potential of Semantic Web technology in the context of agile software development is massive. At the same time this potential is massively underappreciated. The focus of the W3C Software Engineering Task Force group [1] has been on bringing together Semantic Web technologies and Software Engineering. For example, the Task Force's document on Ontology Driven Architecture [2] outlines a role for OWL in OMG's vision of Model Driven Architecture, which is typically classified among the "big modeling upfront", systematic engineering approaches. However, Software *Engineering* is only one aspect of the software development field. Arguably most software is nowadays developed using rather agile, light-weight approaches. Agile methods like unit testing and refactoring have become mainstream years ago, and are much better supported by tools than MDA ideas, which still have to be proven in practice. In related fields such as multi-agent systems there is also evidence [4] that formal design methods are not necessarily the best choice. In an open setting like the Semantic Web these results can likely be applied as well.

Based on my background as a developer of the Protégé-OWL platform [3], I think the Semantic Web community should now address the issue of how to support contemporary software developers with their short-term, real-world problems. Protégé has an active user community, and many of these users are actually developing ontology-based software. These people (and their decision makers) often ask about the purpose of ontology development, and how to have ontologies interact with the rest of their application architecture. For these people our community should explain that OWL and its tools like Protégé have many advantages compared to similar tools from the UML world (e.g., easier to use for end-users, different formal expressiveness with reasoning support, integrated form generation to acquire individuals, built-in test cases with consistency checks, ideal for rapid-prototyping). It would be great to see publications about this appear in main stream computer science journals (JDJ etc). Another main feature of the Semantic Web is to have domain models that are not only used to generate other software artifacts (like in MDA), but also to share and publish the models online. This strengthens reusability. At the same time, OWL and extensions like SWRL can be used at run-time to do reasoning, or even to drive the control logic of a program. In a sense they are executable just like programming code. This means that software moves up to a higher level of abstraction. With Semantic-Web based development, agile approaches can be taken to the extreme, because feedback after a change is available immediately, reasoners can automate testing, and the customer is more directly involved.

Other questions that we need to address are more practical. For example, we should guide users through the jungle of APIs, tools and platforms by means of independent surveys. Programmers also need to understand the relation of these APIs to techniques known from mainstream object-oriented development. For example, Jena and the Protégé-OWL API essentially implement a design pattern called Dynamic Object Model [5]. Accessing ontologies at run time as dynamic models has the advantage that generic algorithms like reasoners can be executed easily. However, such generic classes are on a different level of abstraction than the rest of the code, and for example don't allow for procedural attachment. In many cases it is therefore convenient and clean to have object-oriented source code generated from OWL classes, so that the ontologies can be seamlessly connected into the remaining code (user interface etc). The generated classes will extend the generic API classes, so that objects are at the same time dynamic yet integrated with the remaining object-oriented code. We should provide recommendations on code generation such as Java templates, based on implementations like Kazuki or the Protégé-OWL API code generator. Software architectures based on this idea no longer require UML at all – the Java classes are directly generated from the domain model and only serve as a wrapper of active objects that are in fact hybrids of OWL and Java. Things that cannot be expressed in pure Java just remain in the OWL "view", while procedural aspects of the model are coded in the Java "view".

I think what have in front of us is not only an extension of Model-Driven Architecture. We are in fact talking about a different development paradigm, which is attractive to many domains in the Semantic Web and beyond. The links into MDA are important and valuable on their own, but I think it's time to look into how to connect all this with other mainstream technologies.

References

[1] http://www.w3.org/2001/sw/BestPractices/SE/
[2] http://www.w3.org/2001/sw/BestPractices/SE/ODA/
[3] http://protege.stanford.edu/plugins/owl/
[4] Bruce Edmonds, Joanna Bryson. The Insufficiency of Formal Design Methods - the necessity of an experimental approach, AAMAS 2004
[5] Dirk Riehle, Michel Tilman, Ralph Johnson. Dynamic Object Model. *Pattern Languages of Program Design 5.* Edited by Dragos Manolescu, Markus Völter, James Noble. Reading, MA: Addison-Wesley, 2005